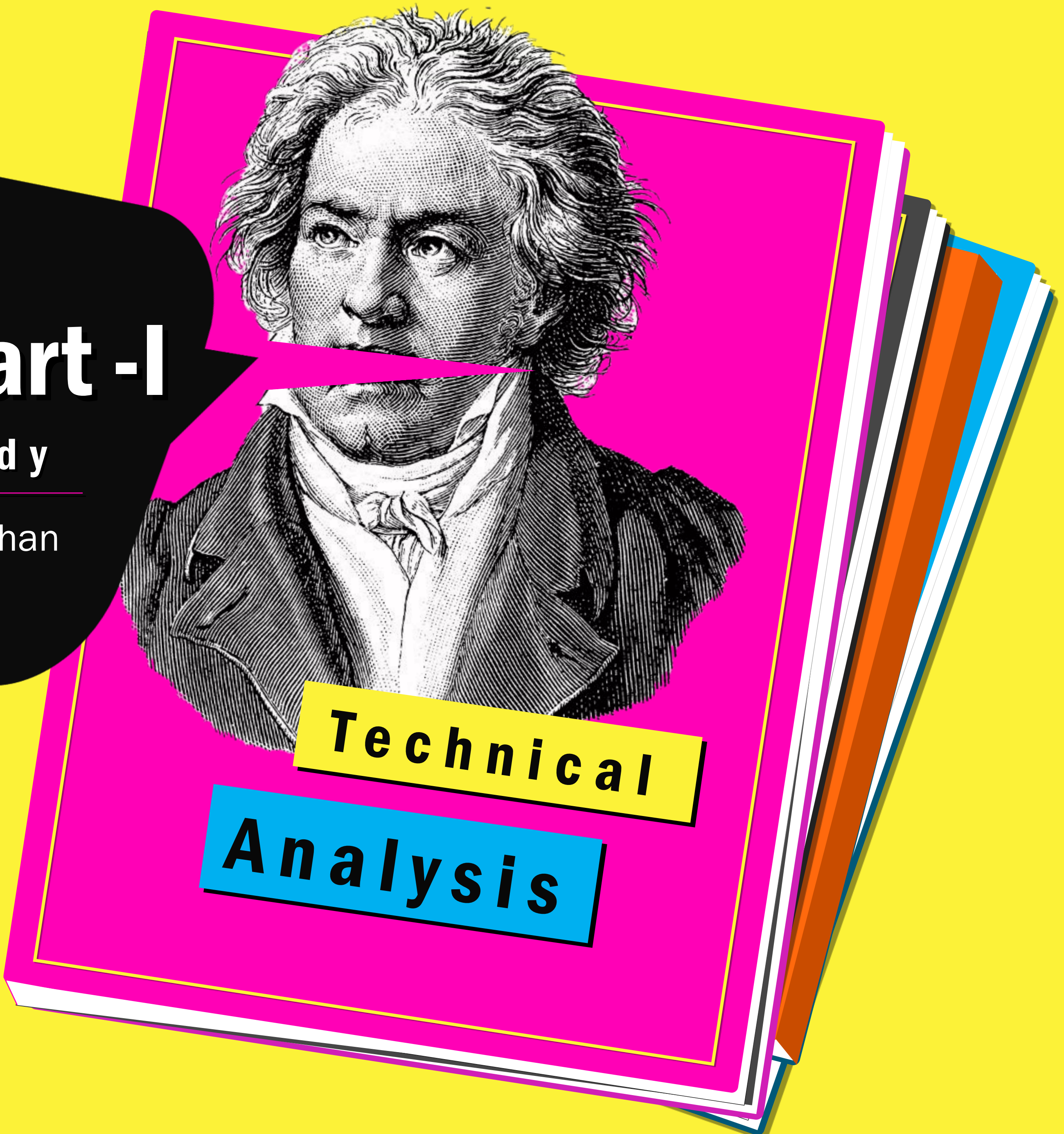
Node JS Part -I

A descriptive study

By A S Padmanabhan



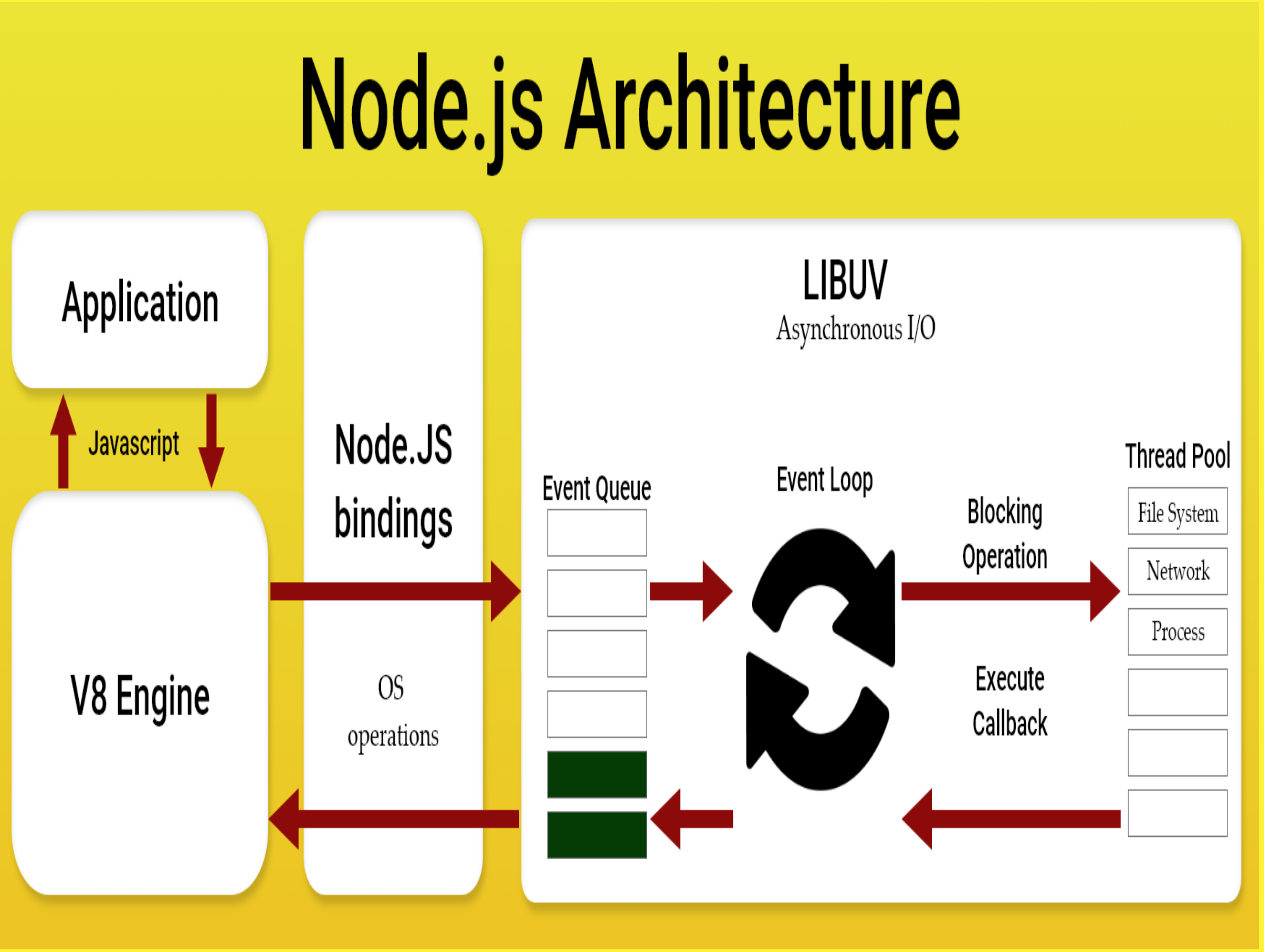
Node JS Introduction

What is Node JS ,what are architectural details ,how things work inside node js

A S Padmanabhan

What is Node JS

- Node.js is an open-source and cross-platform JavaScript runtime environment
- It runs on V8 JavaScript engine, the core of Google Chrome, outside of the browser
- Application in Node Js are single threaded applications
- Node JS uses a single event driven non-blocking I/O model
- V8 is Google's open source high-performance JavaScript and WebAssembly engine, written in C++.
- It is used in Chrome and in Node.js,
- It implements ECMAScript and WebAssembly, and runs on Windows 7 or later, macOS 10.12+, and Linux systems that use x64, IA-32, ARM, or MIPS processors
- V8 can run standalone, or can be embedded into any C++ application

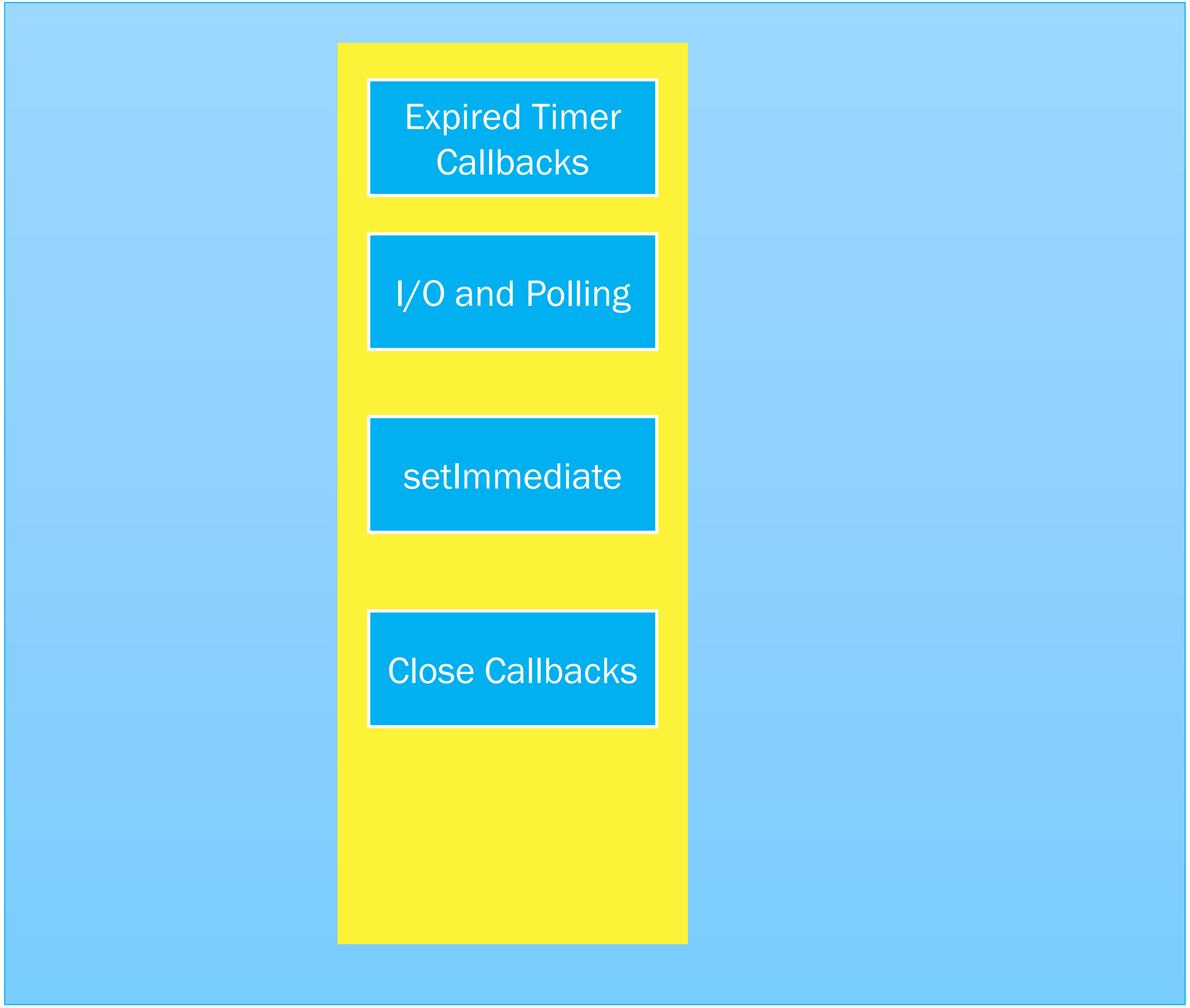


Node JS Architecture

- Libuv gives access to underlying os ,filesystem and network system
- Libuv implements event loop and thread pool
- event loop is responsible for light weight tasks like call back functions or network i/o
- thread pool is responsible for heavy task like file access ,compression
- Libuv is written in c++

Event Queue Phases

- Expired Timer Callbacks
- I/O Polling and Callbacks
- setImmediate callbacks
- Close callbacks



Functions and Modules in Node js

Creating asynchronous functions ,working with Promise object and creating modules

A S Padmanabhan

Async function in node js

- Synchronous operations in javascript are executed one at a time
- Aysnchronous operations run at the same time ,that is parallely to other synchronous operations
- An asynchronous operation can be created using callbacks
- Another way to create async operations is through Promises
- Promise are returned as objects from a function declared as async
- A Promise is a proxy for a value not necessarily known when the promise is created.
- It allows you to associate handlers with an asynchronous action's eventual success value or failure reason. This lets asynchronous methods return values

- like synchronous methods: instead of immediately returning the final value, the asynchronous method returns a promise
- to supply the value at some point in the future.
- A Promise is in one of these states:
- pending: initial state, neither fulfilled nor rejected.
- fulfilled: meaning that the operation was completed
- successfully.
- rejected: meaning that the operation failed.

Modules in node js

- Every file in node application is considered a module
- Functions written in file are private to this module
- To access them outside module they need to be exported
- There are two module systems in node js ,these are CommonJS and ECMAScript module

Using Common JS modules

```
function calculate(a,b)
{
    return a+b
}
function multiply(a,b)
{
    return a*b
}
module.exports={calculate,multiply}
```

```
const mod=require('./module1.js’)
const res=mod.multiply(12,34)
console.log(res)
```

Enabling ECMAScript modules

- To enable ECMAScript modules ,an key “type” with value “modules” has to be added to package .json

```
{
  "name": "ecmapdemo",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\"
&& exit 1"
  },
  "author": "",
  "license": "ISC",
   "type":"module“
}
}
```

A S Padmanabhan

USING ECMAScript module

```
function calculate(a,b)
{
    return a+b
}
function multiply(a,b)
{
    return a*b
}
export  {calculate,multiply}
```

```
import {calculate,multiply} from
'./module1.js'

console.log(calculate(12,23))
console.log(multiply(12,23))
```

Built In Modules in Node JS

- Event
- Fs
- Http/Https
- Net
- Os
- Path
- Timer
- Url

Events module

- Events Module is used to work with events
- All events are instance of EventEmitter object

```
import events from "events"
import fs from 'fs'
var eventEmitter=new events.EventEmitter()
eventEmitter.addListener("write-to-console",()=>{
    console.log("Hi There")
})
eventEmitter.addListener("write-to-file",()=>{
    fs.writeFile('data.txt','Hello World',()=>{

    })
})
```

FS module

- Fs module is used to work with file system of computer

```
import fs from 'fs'
fs.writeFile('data1.txt',"Writing data to file",(err)=>{
   if(err) console.log(err.code,err.errno,err.syscall)
})
fs.readFile("data1.txt",'utf8',(err,data)=>{

    if(err) console.log(err)
    else
     console.log(data)
})
```

-

Http and Https modules

- Http and https modules can be used to create an http and https server

```
import http from 'http'
const server=http.createServer((req,res)=>{
    res.write("Hello from server")
    res.end()
})
server.listen(5050)
```

Net Module

- The net module provides an asynchronous network API for creating stream-based TCP or IPC servers and clients

```
import net from "net"
const server=net.createServer((connection)=>{
    console.log("client connected")
    connection.write("message to client")

})

server.listen(5000,()=>{
    console.log("server listening")
})
```

```
import net from 'net'
var client=net.connect({port:5000,host:"localhost"},()=>{
    console.log("connected to server")
})
client.on('data',(data)=>{
    console.log(data.toString())
    client.end()
})
```

OS Module

- The OS module provides information about the computer's operating system.

```
import os from 'os'

console.log("System Architecture ",os.arch())
const cpus=os.cpus()
for(let i=0;i<cpus.length;i++)
{
    console.log(cpus[i].model)
    console.log(cpus[i].speed)
    console.log(cpus[i].times)
}

console.log("Free Memory ",os.freemem())
console.log("Total Memory ",os.totalmem())
console.log("Host Name ",os.hostname())
```

Path Module

- Path module provides a way to work with directories and path

```
import path from 'path'
const filepath="d:/nodeapps/myfile.txt"
var filename=path.basename(filepath)
console.log(filename)
console.log(path.delimiter)
console.log(path.dirname(filepath))
console.log(path.extname(filepath))
console.log(path.isAbsolute(filepath))
```

Timers module

- clearImmediate() cancels setImmediate function operation
- clearInterval()   cancels setInterval function operation
- clearTimeout()   cancels setTimeout function operation
- ref() makes timeout object active
- unref() makes timeout object inactive
- setImmediate() executes a given operation immediately
- setInterval() executes a given operation repeatedly based on time given
- setTimeout() executes a given operation after a period of given time

Url module

- The URL module splits up a web address into readable parts.

```
import url from 'url'
const urltext="http://localhost:8080/myapp/home.html?item=coffee&price=200"
const myurl=url.parse(urltext,true)
console.log(myurl)
```